interface. As long as each implementation accesses the other implementations through the meta-implementation layer, different implementations can be mixed together even if they are on different platforms, written in different languages, and using different implementation paradigms (like relational database structures versus object-oriented structures versus structured programming).

[0198] An accessor of the present invention provides access to an implementation in a way that is consistent with the descriptor of that implementation. There is always a one-to-one mapping between a descriptor and an accessor. There is not necessarily a one-to-one mapping between an accessor and an implementation. The accessors of the present invention are interfaces an accessor must implement to participate in the meta-implementation. Different meta-implementations may implement the same accessor interface differently. Accessors have a one-to-one association relationship with descriptors for which this accessor provides access to an implementation. Accessors have a one-to-many association relationship with implementations that corresponds to the features described in the descriptor. The accessor may use parts of one or several different implementations in order to provide access in a manner consistent with the descriptor. An accessor has a one-to-one association relationship with a name that is the name of the accessor. An accessor has a zero-to-one association relationship with a description that provides details about the hint for the correct use of this accessor and details about the implementation. A description is useful for human users and automated documentation. This description is different from the description on the metamodel object. That description provides details about how the model is designed and why; this description provides details on use of the implementation (as used through the accessor). An accessor may produce signals related to events occurring in the accessor implementation. Listeners interested in receiving these signals may use registerImplementationListener(listener) to register interest. Listeners no longer wishing to receive these signals may use deregisterImplementationListener(listener) to stop receiving signals.

[0199] An accessor of the present invention provides access to an implementation. Use of the implementation depends of the submodel of accessor. A meta-implementation change event is fired whenever an attribute value is added, changed or removed from the accessor.

[0200] A failure descriptor in the metamodel layer describes the failures an operation throws, not the failure itself. The failure is modeled using a metamodel. To access a failure, use the metamodel accessor for the failure. Otherwise, the tool using the accessor that generates the failure must handle these failures.

[0201] Constraints are not accessed via an accessor to an implementation. Instead the implementation provides the constraints or a constraint implementation is added to the accessor making use of the constraint. The accessor imposes the restriction, even if it is not imposed by the true implementation.

[0202] A feature accessor of the present invention is an accessor that is part of a larger accessor. A feature accessor has a one-to-one association relationship with a feature descriptor for which this accessor provides access to an implementation. A feature accessor has a one-to-many association relationship with an implementation that corresponds to the features described in the descriptor. A feature accessor may use parts of one or several different implementations in order to provide access in a manner consistent with the descriptor. A feature accessor has a zero-to-one association relationship with a parent accessor that is a reference to the accessor containing the feature accessor. A feature accessor has a zero-to-many association relationship with an access constraint implementation that enforces restrictions related to security access.

[0203] A feature accessor provides access to an implementation. Use of the implementation depends of the submodel of feature accessor. A feature accessor adds no additional events.

[0204] A datatype accessor of the present invention is an accessor that provides access to data. Attribute Accessor and Parameter Accessor inherit from datatype accessor. A datatype accessor has one-to-one association relationship with a datatype descriptor that is the attribute descriptor for which this accessor provides access to an implementation. A datatype accessor has one-to-one association relationship with a datatype implementation that is the attribute implementation that this accessor uses to perform the actual storage and retrieval of values. A datatype accessor has zero-to-many association relationship with occurrence constraints that are each an implementation of a constraint to restrict the number of values assigned to the attribute implementation. A datatype accessor has zero-to-many association relationship with value constraints that are an implementation of a constraint to restrict the values allowed to be held by an attribute implementation. A datatype accessor has zero-to-many association relationship with access constraints that are each implementation of a constraint to restrict the access to values held by the attribute implementation.

[0205] An instance of a datatype accessor has the following may include the following operations a getValue( ) that returns the current value(s) after checking access constraints, a setValue(Instance) that removes any previous value and sets it to the new value after checking all constraints, an addValue(Instance) that adds the instance given to the current values after checking all constraints, a removeValue(Instance) that removes the given instance from the current values after checking to see if the value is being held and checking access and occurrence constraints, and clearValues( ) that remove all values after checking access and occurrence constraints.

[0206] A datatype accessor adds no additional events.

[0207] An attribute accessor of the present invention is a feature accessor that provides access to an attribute of a model implementation. Unlike an attribute descriptor, an attribute accessor does not aggregate a datatype accessor. The datatype is accessed directly through the attribute implementation, not a datatype accessor. Attribute accessors allow a value to be set and retrieved. If static, the value of the attribute is shared by all instances. Otherwise the value is assigned only to a single instance.

[0208] An attribute accessor has a one-to-one association relationship with an attribute descriptor that is the attribute descriptor for which this accessor provides access to an implementation. An attribute accessor has a one-to-one